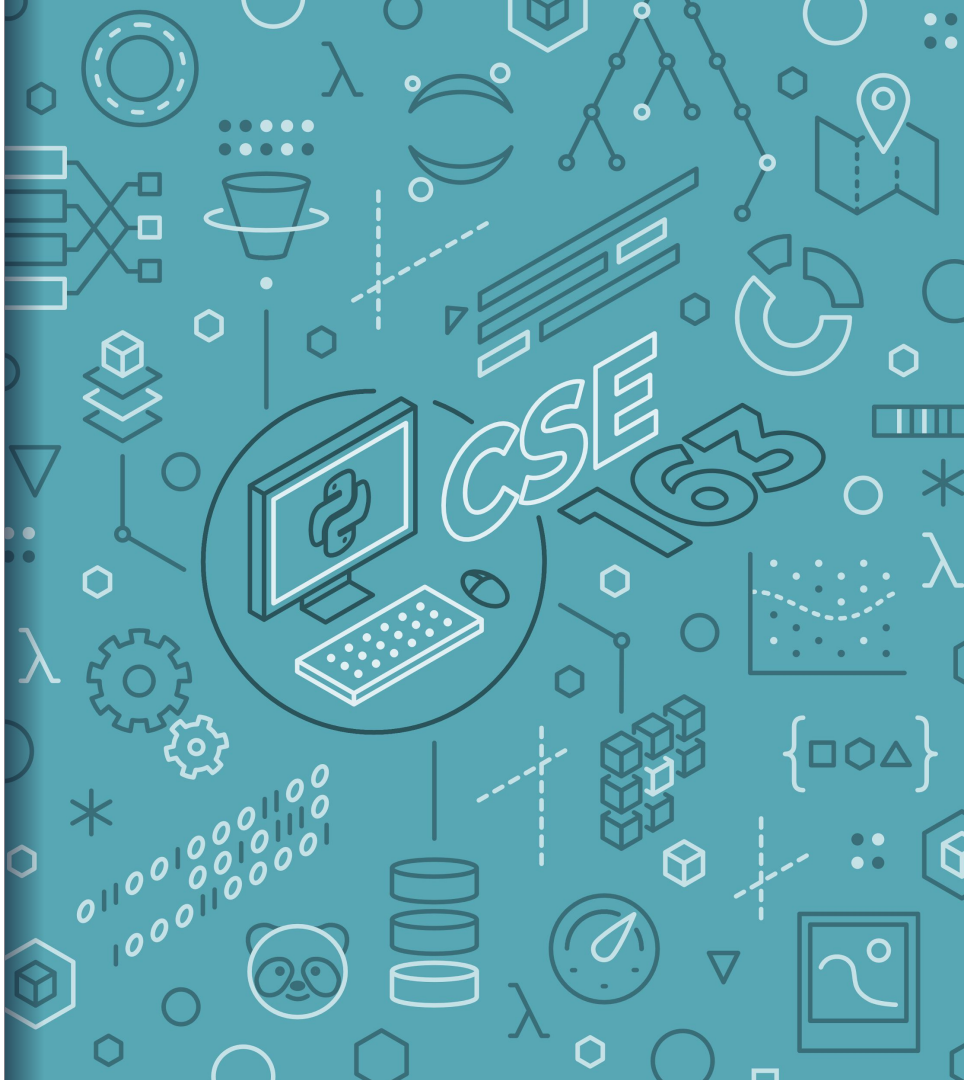
CSE 163

Lists and File Processing

Hunter Schafer



Last Time Time

- Jupyter Notebooks
- Python Crash Course - Day 2
  - Types + Casting
  - Loops
  - Conditionals
  - Functions (parameters + returns)
  - Strings

This Time

- Lists
- Working with Files
- Documenting Code
- None
- Homework Logistics
  - Testing code

Strings

```
s = 'hello world'
# Length
len(s)  # 11

# Indexing
s[1]           # 'e'
s[len(s) - 1]  # 'd'

# Looping
for i in range(len(s)):
   print(s[i])

for c in s:
    print(c)
```

Slices

h	e	l	l	o		w	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

- You are able to index to get more than one value

```
s[0:2]          # 'he'
s[4:len(s)]     # 'o world'

s[4:]           # 'o world'
s[:len(s) - 2]  # 'hello wor'

s[0:8:2]        # 'hlow'
s[::-2]         # ‘drwolh’
```

General Syntax: `start:stop:step`

Negative Indices?

h	e	l	l	o		w	o	r	l	d
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- Python also allows you to use negative numbers for indices to start from the end of the string!
- Instead of saying

```
s[:len(s) - 2]
```

- You would instead say

```
s[:-2]  # Not the same as s[::-2]
```

String Functions

- Use useful functions you can call on string objects
  - Casing: `upper, lower`
  - Find Location: `find, index`
  - Remove Whitespace: `strip, lstrip, rstrip`
  - Breaking/Building Strings: `split, join`

Lists

- A List is a generic **collection** that holds multiple values
  - Each value has an index
  - String is like a list of length 1 strings
- A list can store multiple values of any types

```
l1 = [1, 2, 3, 4]
l2 = ['hello', 'goodbye']
l3 = [1, 'dog', 3.4]
```

- See List Demo

Strings

vs.

Lists

```
s = 'hello world'
# Length
len(s)  # 11

# Indexing
s[1]           # 'e'
s[len(s) - 1]  # 'd'

# Looping
for i in range(len(s)):
   print(s[i])

for c in s:
    print(c)
```

```
l = ['dog', 'says', 'woof']
# Length
len(l)  # 3

# Indexing
l[1]           # 'says'
l[len(l) - 1]  # 'woof'

# Looping
for i in range(len(l)):
   print(l[i])

for word in l:
    print(word)
```

Brain Break

Files

- A **file** is data stored on computer. This data can represent almost anything (Word document, picture, song, etc.)!
- For the next two weeks, we will focus on files that store **text** data (often called *plain-text*). An example:

```
cat poem.txt
```

```
she sells
sea
shells by
the sea shore
```

Files in Python

- See File Demo
- Use the open function to open a file. Syntax is a bit weird

```
with open('poem.txt') as file:
    print(file)

# <_io.TextIOWrapper name='poem.txt' mode='r' encoding='UTF-8'>
```

- Instead, you have to ask the file to actually read itself

```
with open('poem.txt') as file:
    print(file.read())

# Prints file contents
```

General Patterns

- Process a file line by line

```
with open(file_name) as file:
    lines = file.readlines()
    for line in lines:
        # do something with line
```

- Process each word on a file in a line

```
with open(file_name) as file:
    lines = file.readlines()
    for line in lines:
        words = line.split()
        for word in words:
            # do something with word
```

None

- None is a special value in Python that represents the absence of a value
- You don't have to know a lot (i.e. none) about None except:
  - It can cause your program to crash if you ask a None value to do something
  - You commonly will return None in bad cases

```
def increment(x):
    if x < 0:
        return None
    else:
        return x + 1

if increment(-1) is None:
    print('Failed')
```

Homework Logistics

- Due next Thursday at 11:59 pm
  - Submit code on Gradescope
  - Submit reflection on Google Forms
- **Start Early! Start Early! Start Early! Stary Early! Start Early! Start Early! Start Early! Stary Early! Start Early! Start Early! Start Early! Stary Early! Start Early! Start Early! Start Early! Stary Early! Start Early! Start Early! Start Early! Stary Early! Should I wait until Wednesday to start? No! Start Early! Start Early! Stary Early! Start Early! Start Early! Start Early! Stary Early! Start Early! Start Early! Start Early! Stary Early! Start Early! Start Early! Stary Early! Start Early! Start Early! Start Early! Stary Early! Start Early! Start Early! Start Early! Stary Early! Start Early! Start Early! Start Early! Stary Early! Start Early!**

Next Time

- Advanced List Operations
- Sets
- Tuples

Before Next Time

- Keep up with practice!
- Start the assignment!